Find Me a Hash

Susan Landau

We're accustomed to hearing about the unreasonable effectiveness of mathematics, delightful—and unexpected—applications of theory to the real world. In the world of the Internet, we've seen it in the use of number theory in public-key cryptography (the Diffie-Hellman system, the RSA algorithm, elliptic curve cryptosystems), in the utilization of graph theory in network design. In the world of Internet data security, currently we face the opposite situation: a problem in search of mathematical theory. The problem is hash functions.

A hash function is an easy-to-compute compression function that takes a variable-length input and converts it to a fixed-length output. The hashes in which we are interested, called cryptographic hash functions, are "one-way", which is to say, they should be easy to compute and "hard", or computationally expensive, to invert[1]. Hash functions are used as a compact representation of a longer piece of data—a *digital fingerprint*—and to provide message integrity. The way hashes are used to provide integrity is that the hash value of a particular piece of data, h_0, is computed at an initial time t_0. When the data needs to be used later at time t_1, the hash, h_1, is recomputed. If the two hashes are equal, then the data has not been altered. Ralph Merkle, a co-inventor of public-key cryptography, calls hashes the "duct tape" of cryptography. Among other things, hashes are used to ascertain software integrity, in digital signatures, in message authentication, and as one-time passwords; they are employed in many Internet protocols including SSL/TLS, the transport-layer protocol that enables secure Web transactions, IPsec, and SSH.

Because hash functions "shrink" data, collisions between hashes are inevitable. There are three fundamental properties that a cryptographic hash should satisfy: *pre-image resistance* (sometimes called *non-invertibility*): it should be computationally infeasible to find an input which hashes to a specified output, *second pre-image resistance*: it should be computationally infeasible to find a second input that hashes to the same output as a specified input, and *collision resistance*: it should be computationally infeasible to find two different inputs that hash to the same output. In 1979 Merkle [10, pp. 12–13] and Gideon Yuval [12] independently observed that because of the "birthday" paradox—the well-known result that in a group of twenty-three people, the probability that two people share the same birthday is slightly more than half—on average one needs to search only square root of the search space to find a collision. Thus hash functions of n bits are at best $n/2$ bits secure against collision attacks.

A common method for hash function design is iteration: use a small set of operations combined in what is called a *round*. This way the algorithm is put together using a small set of instructions, an important issue for implementations (the same design paradigm is also true for block ciphers). Merkle and Damgård independently showed how to take a collision-resistant compression function and iterate it to produce a collision-resistant hash

Editor's Note: The month of April is Mathematics Awareness Month. The 2006 theme for MAM is Mathematics and Internet Security.

Susan Landau is at Sun Microsystems. Her email address is `susan.landau@sun.com`.

[1] *Hash functions may be "keyed" or "unkeyed" depending on the application; we will ignore that distinction here, although we note that keyed hash functions include Message Authentication Codes, or MACs, whose theory is well understood.*

function [11], [2]. The idea, called the Merkle-Damgård construction, goes as follows: Let f be a collision-resistant compression function that maps $n+m$ bits to n bits. Break the input I into blocks $x_1, \dots, x_k$ of length n, padding out the last block with 0's as necessary. Start with a random string of n bits, called the initialization vector, IV. Let b be the length of I and define an extra block x_{k+1} which holds the right-justified representation of b written in binary (assume that $b < 2^n$); the left part of x_{k+1} is filled with 0's. Then the hash function h is defined by: $h(x) = H_{k+1}$ where $||$ denotes concatenation and:

$$H_0 = IV$$
$$H_i = f(H_{i-1}||x_i), 1 \le i \le k+1.$$

That h is collision-resistant follows from the observation that any collision for h would imply a collision for f at some stage i.

The Merkle-Damgård construction underlies most popular hash functions. A half-dozen years ago, there were several popular cryptographic hash functions from which to choose, including MD5 and SHA-1. MD5, developed by Ron Rivest, is a 128-bit hash that is a strengthened form of an earlier Rivest hash function, MD4. Because of the birthday attack, MD5 can only be 64 bits strong; the short length contributed to the National Institute of Standards and Technology (NIST) decision not to certify MD5 as a Federal Information Processing Standard (FIPS). FIPS certification means the system is approved for sale to the federal government, an important market for the computer industry. Instead NIST certified SHA-0, the Secure Hash Algorithm (SHA-0 was originally simply SHA). The algorithm was developed by the National Security Agency (NSA) and was also based on MD4; it was a 160-bit hash that was built to work with Skipjack, the block cipher with 80-bit key that was part of the Clipper system. Two years later NSA proffered SHA-1, an algorithm that differed from SHA-0 by a circular shift operation in the round function, which also became a FIPS. The rapid development of SHA-1 shortly after SHA-0 caused users to shy away from the earlier algorithm—and indeed weaknesses were found in SHA-0—but both MD5 and SHA-1 have been widely deployed (MD5 was used in many protocols despite not being a FIPS). And as a result we are in trouble.

MD5 was already in difficulty in 1993, when Bert den Boer and Antoon Bosselaers found problems with its compression function [3]; further problems were discovered three years later by Hans Dobbertin [4]. The situation became a great deal worse in 2004. At a cryptography meeting in Santa Barbara, California, Xiaoyun Wang, Denggou Feng, Xuejia Lai, and Hongbo Yu received a standing ovation for work showing collision attacks on MD5 (the attacks also applied to several other hash functions: HAVAL, MD4, and RIPEMD) [13]. There had already been a move away from MD5, but this was the final blow. At the same meeting, Eli Biham and Rafi Chen [1] showed how to find "near" collisions in SHA-0 and Antoine Joux demonstrated an actual collision attack on SHA-0 [5]. SHA-1 still seemed safe.

In 2005 the situation got worse. Wang, in collaboration with Yiqun Lisa Yin and Hongbo Yu, showed a collision attack on SHA-1 that took 2^{69} steps (instead of the expected 2^{80}) [14]; then Wang, in collaboration with Andrew Yao and Frances Yao, demonstrated a collision attack on SHA-1 that required only 2^{63} steps [15]. The good news is that this was not a second pre-image attack and the attack does not mean that all protocols using SHA-1 for integrity were at risk (for example, the usage of SHA-1 in the "handshake" of SSL 3.0/TLS protocol is not affected by these attacks). But it did mean that SHA-1 should be replaced as quickly as possible and should not be used for new applications. How to proceed was both clear and cloudy.

SHA-256, also developed by NSA is waiting in the wings, and is already a FIPS (as are SHA-384 and SHA-512). SHA-256 is three to four times slower than SHA-1, but that is not the real problem. Moving a new algorithm into the infrastructure, whether SHA-256 or a direct SHA-1 replacement (including the established RIPEMD-160, SHA-256 truncated to 160 bits, or a "patched" SHA-1), is not an easy task. Although computer manufacturers understand the importance of replacing SHA-1, and SHA-256 is in the next operating systems being fielded by Microsoft, Sun, and other manufacturers, SHA-1 and MD5 will remain in legacy systems for years to come. And while SHA-256 may share some of the structure of SHA-1 and thus be potentially vulnerable to attack, at 256 bits, the algorithm is large enough, and strong enough, to suffice for now. This is the clear step forward.

The cloudy part is what happens next. Ten years ago, there was a need to replace the Data Encryption Standard (DES), the encryption algorithm with a 56-bit key that had been functioning since the 1970s (see, for example, [6], [7]). Fortunately, since the 1980s there had been fundamental research into the design of block ciphers, much of it from ideas learned through attacks on DES. The strength of the Advanced Encryption Standard (AES), approved as a FIPS in 2002, is based on that research [8]. It is clear that we need new hash functions, but hash research is not in the same place as block ciphers were a decade ago. Until we really understand the underpinnings of secure hash functions[2], it does not make sense for NIST to begin a serious competition for the next one. DES, a good algorithm for

[2] *There are hash functions based on hard mathematical problems, making them likely to be secure, but these hash functions are inefficient and not used in practice.*

developing an understanding of the security of block-structured ciphers, provided practice for cryptanalysts. SHA-256 is probably the right place to start to do the same for secure hash functions. What is the theory of hash functions? It is not often that mathematicians are asked to develop a theory for duct tape, but there is a clear and present need to do so now for cryptographic hash functions.

References

[1] E. Biham, R. Chen, A. Joux, P. Carrbault, W. Jalby, and C. Lemuet, Collisions in SHA-0 and Reduced SHA-1, *Advances in Cryptology—Eurocrypt '05,* pp. 36–57, 2005.

[2] I. B. Damgård, A Design Principle for Hash Functions, *Advances in Cryptology—CRYPTO '89,* Springer-Verlag LNCS 455, pp. 416–427.

[3] Bert den Boer and Antoon Bosselaers, Collisions for the Compression Function of MD5, *Advances in Cryptology, Proceedings Eurocrypt '93,* Springer-Verlag LNCS 765, 1994, pp. 293–304.

[4] Hans Dobbertin, Cryptanalysis of MD5, Rump Session, Eurocrypt 1996.

[5] Antoine Joux, Collisions for SHA-0, Rump Session, CRYPTO '04, August, 2004.

[6] Susan Landau, Standing the test of time: The Data Encryption Standard, *Notices of the American Mathematical Society,* March 2000, pp. 341–349.

[7] ____ , Communications security for the twenty-first century: The Advanced Encryption Standard, *Notices of the American Mathematical Society,* April 2000, pp. 450–459.

[8] ____ , Polynomials in the nation's service: Using algebra to design the Advanced Encryption Standard, *American Mathematical Monthly,* February 2004, pp. 89–117.

[9] Alfred Menezes, Paul van Oorschot, and Scott Vanstone, *Handbook of Applied Cryptography,* CRC Press, 2001.

[10] Ralph Merkle, *Secrecy, Authentication, and Public Key Systems,* UMI Research Press, Ann Arbor, Michigan, 1979.

[11] ____ , A fast software one-way hash function, *Journal of Cryptology* **3** (1979), 43–58.

[12] Gideon Yuval, How to swindle Rabin, *Cryptologia* **3** (1979), 187–190.

[13] Xiaoyun Wang, Dengguo Feng, Xuejia Lai, and Hongbo Yu, Collisions for Hash Functions, Rump Session, CRYPTO '04, August, 2004.

[14] Xiayun Wang, Yiqun Lisa Yin, and Hongbo Yu, Finding Collisions in the Full SHA-1, *Advances in Cryptology—CRYPTO '05,* pp. 17–36.

[15] Xiayun Wang, Andrew Yao, and Frances Yao, New Collision Search for SHA-1, Rump Session, CRYPTO '05, August 2005.